CIRCUITOS LÓGICOS

Relatório V

Discente: Rebeca de Macêdo Ferreira. Matrícula: 2016000524.

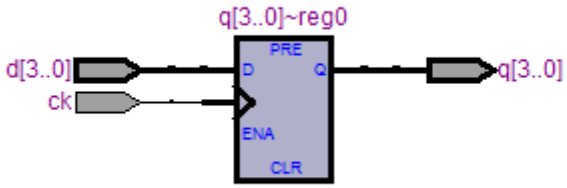
Flip-Flop

Esse é um projeto referente ao circuito flip-flop, iniciamos copiando o código vhd sugerido no livro Harris & Harris.

```
1   library IEEE; use IEEE.STD_LOGIC_1164.all;
2
3   entity flop is
4       port (ck: in STD_LOGIC;
5             d:  in STD_LOGIC_VECTOR (3 downto 0);
6             q:  out STD_LOGIC_VECTOR (3 downto 0));
7   end;
8
9   architecture synth of flop is
10  begin
11      process (ck) begin
12          if ck'event and ck = '1' then
13              q <= d;
14          end if;
15      end process;
16  end;
```

‘ck’ (clock flip-flop) e ‘d’ são as entradas do circuito, portanto ‘q’ é a saída que é determinada pelo ck.

Depois de compilado no Quartus e corrigido pequenos erros, obtivemos como resultado do RTL viewer o seguinte:



Para fazer as simulações no Modelsim construimos um testbench e também a tabela verdade pra nosso caso.

```
7   entity testbench_flop is -- no inputs or outputs
8   end;
9   architecture sim of testbench_flop is
10      component flop
11          port(ck: in STD_LOGIC;
12               d:  in STD_LOGIC_VECTOR (3 downto 0);
13               q:  out STD_LOGIC_VECTOR (3 downto 0));
14      end component;
15
16      signal cktest: STD_LOGIC;
17      signal ck: STD_LOGIC;
18      signal d, q: STD_LOGIC_VECTOR(3 downto 0);
19      signal qexpected: STD_LOGIC_VECTOR(3 downto 0);
20      constant MEMSIZE: integer := 6;
21      type tvarray is array (MEMSIZE downto 0) of
22      STD_LOGIC_VECTOR (8 downto 0);
23      signal testvectors: tvarray;
24      shared variable vectornum, errors: integer;
25      begin
26      -- instantiate device under test
27      dut: flop port map (ck, d, q);
28      -- generate clock
29      process begin
30          cktest <= '1'; wait for 10 ns;
31          cktest <= '0'; wait for 5 ns;
32      end process;
```

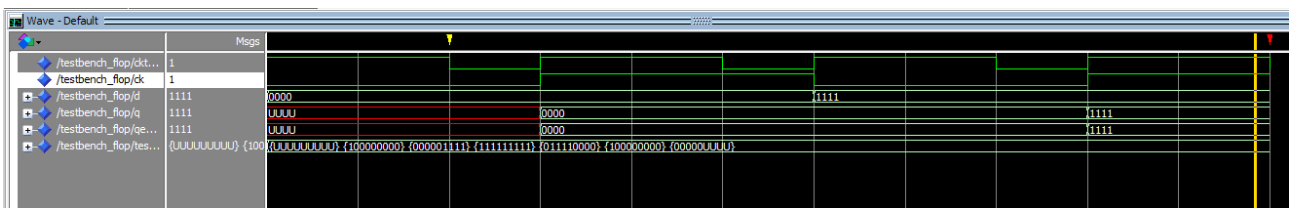
Testbench do flop

```
1   00000_UUUU
2   10000_0000
3   01111_0000
4   11111_1111
5   00000_1111
6   10000_0000
```

Tabela verdade do flop

Na tabela os primeiros números de cada linha corresponde ao valor do clock. Em flip-flops com clock, tendo o clock nível lógico ‘1’ o valor de ‘d’(entrada) é copiado para ‘q’(saída); caso o clock tenha nivél lógico ‘0’ o valor de ‘d’(entrada) não é copiado pra ‘q’(saída), portanto, caso seja a primeira linha de teste o valor da saída será indeterminado.

Simulamos o RTL level:

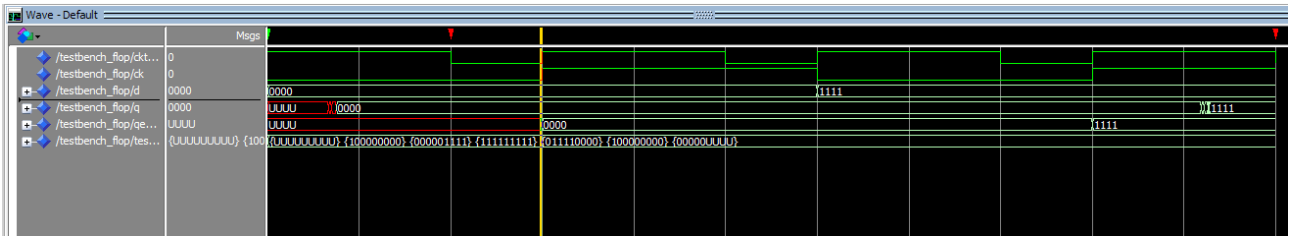


simulação RTL level

```
Transcript
#    Time: 10 ns  Iteration: 1  Instance: /testbench_flop
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#    Time: 10 ns  Iteration: 1  Instance: /testbench_flop
# ** Failure: Just kidding --4tests completed successfully.
#    Time: 55 ns  Iteration: 1  Process: /testbench_flop/line__76 File: D:/CIRCUITOS/flop/testbench/testbench_flop.vhd
# Break in Process line__76 at D:/CIRCUITOS/flop/testbench/testbench_flop.vhd line 93
```

Transcript RTL level

Simulamos o Gate level:



Simulação Gate level

Há uma seta vermelha indicando um “erro”, isso se dá pelo fato de no testbench não ter especificado interpretações pra saídas indeterminadas, sendo assim, a seta indica apenas um alerta pra tal situação.

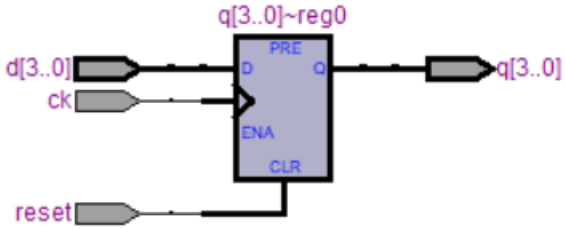
Resettable flip-flop assíncrono

Nesse projeto simulamos um resttable flip-flop assíncrono, ele se difere do flip-flop anterior por possuir uma entrada Reset que pode alterar as saídas. São assíncronos pois fazem o reset independentemente do estado do clock.

O código VHD a seguir foi copiado do livro Harris&Harris e diz respeito ao circuito resettable flip-flop assincrono.

```
1  library IEEE; use IEEE.STD_LOGIC_1164.all;
2
3  entity flopra is
4      port(ck: in STD_LOGIC;
5           reset: in STD_LOGIC;
6           d: in STD_LOGIC_VECTOR(3 downto 0);
7           q: out STD_LOGIC_VECTOR(3 downto 0));
8  end;
9
10 architecture asynchronous of flopra is
11 begin
12     process(ck, reset) begin
13         if reset = '1' then
14             q <= "0000";
15         elsif ck' event and ck = '1' then
16             q <= d;
17         end if;
18     end process;
19 end;
```

Após compilar o código no Quartus obtivemos o RTL viewer:



Então pra fazer simulações com o Modelsim precisamos criar o testbench e fazer um arquivo .tv com as informações da tabela verdade pra o circuito.

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_arith.ALL;
4   use IEEE.STD_LOGIC_unsigned.ALL;
5   use STD.TEXTIO.ALL ;
6
7   entity testbench_flopra is -- no inputs or outputs
8   end;
9   architecture sim of testbench_flopra is
10      component flopra
11          port(ck, reset: in STD_LOGIC;
12               d: in STD_LOGIC_VECTOR(3 downto 0);
13               q: out STD_LOGIC_VECTOR(3 downto 0));
14      end component;
15
16  signal clk: STD_LOGIC;
17  signal ck: STD_LOGIC;
18  signal reset: STD_LOGIC;
19  signal d, q: STD_LOGIC_VECTOR(3 downto 0);
20  signal qexpected: STD_LOGIC_VECTOR(3 downto 0);
21  constant MEMSIZE: integer := 15;
22  type tvarray is array (MEMSIZE downto 0) of
23  STD_LOGIC_VECTOR (9 downto 0);
24  signal testvectors: tvarray;
25  shared variable vectornum, errors: integer;
26  begin
27  -- instantiate device under test
28  dut: flopra port map (ck, reset, d, q);
29  -- generate clock
30  process begin
31      clk <= '1'; wait for 15 ns;
32      clk <= '0'; wait for 5 ns;
33  end process;
```

testbench do resettable flip-flop assíncrono

Vemos aqui que temos como entradas o reset e clock (1bit cada) e o ‘d’ (4 bits) e como saída temos o ‘q’ (4bits).

```
43      i := 0;
44      FILE_OPEN (tv, "./flopra.tv", READ_MODE);
45      while not endfile(tv) loop
46          readline (tv, L);
47          for j in 9 downto 0 loop
48              read (L, ch);
49              if (ch = '_') then read (L, ch);
50              end if;
51              if (ch = 'U') then
52              testvectors (i) (j) <= 'U';
53              end if;
54              if (ch = '0') then
55              testvectors (i) (j) <= '0';
56              end if;
57              if(ch = '1') then
58              testvectors (i) (j) <= '1';
59              end if;
60          end loop;
61          i := i + 1;
62      end loop;
63      vectornum := 0; errors := 0;
64      -- reset <= '1'; wait for 27 ns; reset <= '0';
65      wait;
66  end process;
```

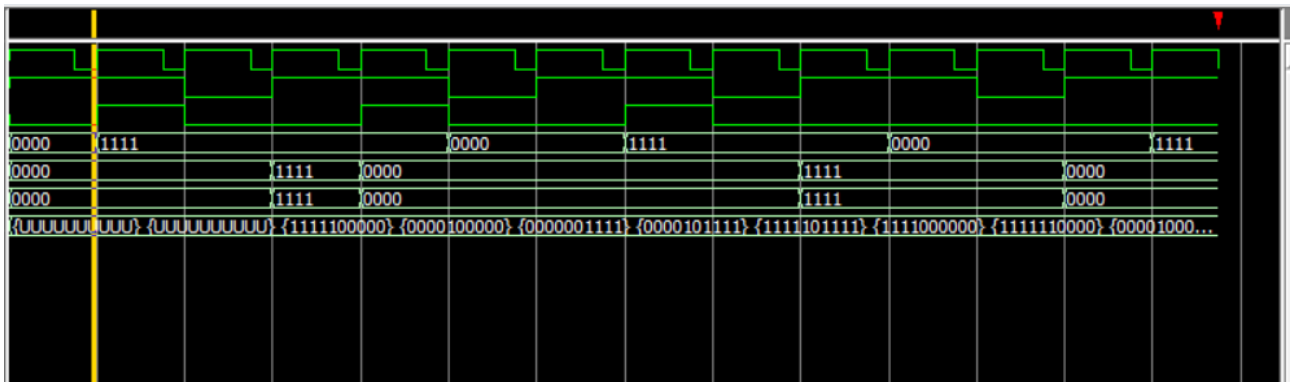
testbench do resettable flip flip assíncrono

```
1   000000_UUUU
2   000010_0000
3   111111_0000
4   111100_0000
5   111110_1111
6   111111_0000
7   000000_0000
8   000010_0000
9   111111_0000
10  111100_0000
11  111110_1111
12  000010_1111
13  000000_1111
14  000010_0000
15  111110_0000
```

tabela verdade flopra

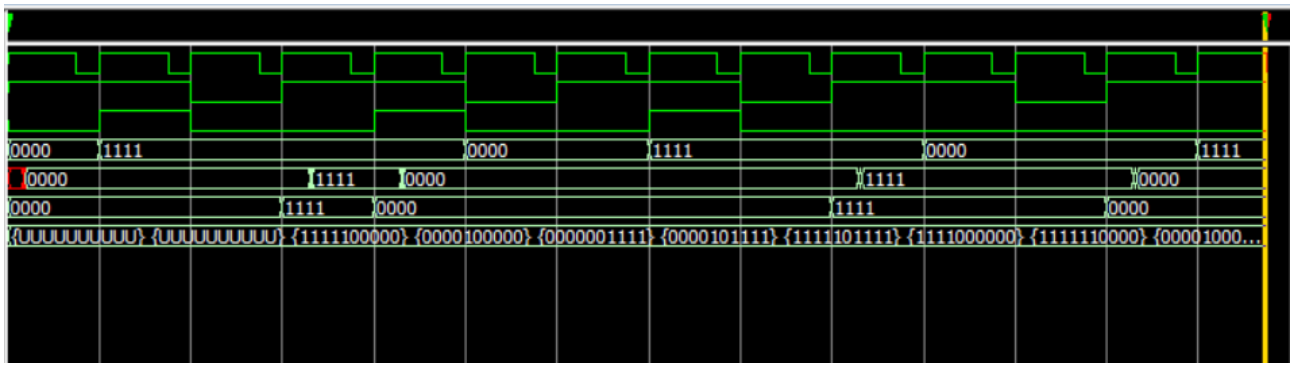
Na tabela verdade acima vemos que a saída ‘q’ do flip-flop resetável é alterada pra 0 (zero), ignorando a entrada ‘d’ , caso o reset tenha nível lógico ‘1’ (true); caso o reset tenha nível lógico 0 (zero) o circuito se comporta com o flip-flip descrito anterior a este. Como se trata de um circuito assíncrono, o clock não tem influência no reset.

Simulamos o RTL level



ondas da simulação RTL level

Simulamos o Gate level



ondas da simulação Gate level

Arvore de diretórios

FLOP:

Código em VHDL do projeto: *flop/ flop.vhd*
Código em VHD do testbench: *flop/ testbench/ testbench_flop.vhd*
Tabela verdade do projeto: *flop/ simulation/ Modelsim/ flop.tv*

FLOPR(assíncrono):

Código em VHDL do projeto: *flopra/ flopra.vhd*
Código em VHD do testbench: *flopra/ testbench/ testbench_flopra.vhd*
Tabela verdade do projeto: *flopra/ simulation/ Modelsim/ flopra.tv*